

A Review of Spring Boot Framework Based on REST API

KARTHIK KUMAR M

Student of Master of Computer Application, Atria Institute of Technology, Bengaluru India
karthikkumarks50@gmail.com

Abstract:

This article gives a general introduction to the Spring framework and How to use it to build REST API-based web applications. Spring is a popular framework renowned for its capabilities and flexibility in creating applications. The report also emphasizes Spring's advantages in terms of architecture and its interoperability with J2EE, among other features and advantages. The emphasis is also on Spring Boot, a Java-based platform that facilitates configuration and deploy Spring applications, especially those that generate REST APIs.

1. INTRODUCTION

Web applications have merged into many different industries in the present digital era, providing consumers with effective services and solutions. Strong frameworks are needed for web application development since they make the process easier and provide scalability, flexibility, and maintainability. Among the many frameworks presently accessible in the software development business, the Spring framework has grown immensely popular.

The Java-based framework Spring offers thorough support for creating enterprise-level applications. It provides a wide range of features and functions, making it suited for several application kinds, including online applications. The dependency injection and inversion of control concepts are upheld by the Spring framework, which also encourages loose coupling and modular architecture.

Spring Boot is a powerful tool for quickly and easily building independent, enterprise-grade apps that is unique within the Spring ecosystem. Convention-over-configuration concepts are used by Spring Boot to reduce boilerplate code

and streamline the development process. By giving options for automatic configuration and skillfully managing dependencies, it simplifies the setup and deployment of applications.

REST (Representational State Transfer) has become a well-liked architectural design approach for creating web services. For client-server communication in contemporary online applications, RESTful APIs have taken over. Developers may create REST API-based web apps quickly and effectively by utilizing the features of the Spring framework.

In this essay, we investigate the significance of the Spring framework and its application to REST API web development.. We go into Spring's features and traits, emphasizing its J2EE compatibility and architectural advantages. Additionally, we concentrate on Spring Boot, a Java-based platform that makes it easier to design and deploy Spring applications, especially when developing REST APIs.

II. SPRING FRAMEWORK ARCHITECTURE

The modular and layered architecture of the Spring framework encourages loose coupling, modularity, and developer-friendliness. To properly utilize the capabilities of the Spring framework, developers must have a thorough understanding of its design. Each layer in the tiered architecture that Spring Boot uses interacts with the one immediately underneath or above it (hierarchical structure).

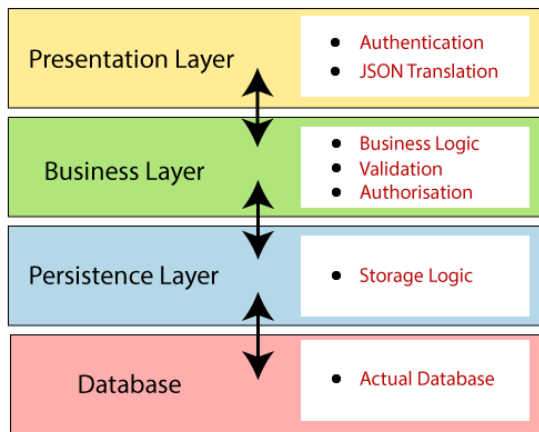


FIG 1. Spring Boot Architecture- layers

Presentation Layer: This layer manages HTTP requests, converts JSON parameters to objects, authenticates the request, and sends it to the Business Layer. It consists mostly of views or the frontend portion.

Business Layer: All business logic is handled by this layer. It utilizes services offered by data access layers and is made up of service classes. It also carries out validation and permission.

Persistence Layer: Business objects are translated from and into database rows by the persistence layer, which also houses all the storage logic.

Database Layer: CRUD (create, retrieve, update, delete) actions are carried out at the database layer.

- *The client sends the server an HTTP request, such as GET, PUT, or POST:*

The flow begins when a client, such as a web browser or mobile application, sends an HTTP request to the server hosting the Spring Boot application. The HTTP method, request headers, and request content (if required) are just a few examples of the various types of information that might be included in the request.

- *The Controller receives the request and translates it to a certain endpoint:*

In Spring Boot, it is up to the Controller to manage incoming requests and allocate them to the appropriate methods in the controller class. The mapping is often specified using annotations like `@RequestMapping`, `@GetMapping`, `@PostMapping`, etc. The endpoint mapping is determined by the request's URL, request parameters, HTTP method.

- *After processing the request, the controller calls the relevant server logic.*

The request is processed by the Controller by carrying out the relevant logic once it selects the appropriate endpoint technique. The logic may involve operations like input verification, computation, data retrieval from databases, or service calls. To take use of route variables or query parameters in the server logic, the Controller may also take data and then extract the information from the request.

- *Using information from the database, the server logic, which is housed in the service layer, conducts business operations:*

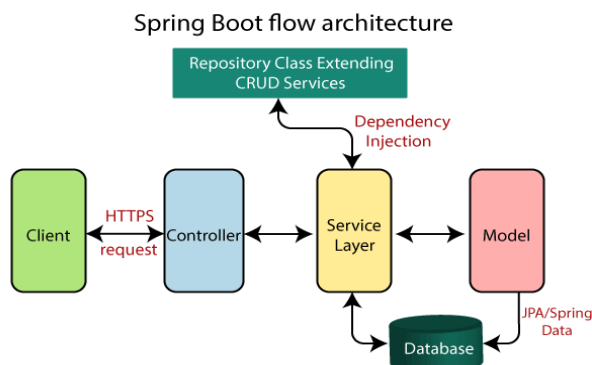


FIG 2.Spring Boot flow architecture

The application's business logic, which includes complicated activities and business rules, is included in the service layer. It often contains reusable and domain-specific functionality that has no direct connection to the web layer. To carry out its functions, the service layer may communicate with other elements like repositories or external services. Data transfer objects (DTOs) are frequently used to move data between layers and guarantee the segregation of concerns.

- *Through the Persistence layer, the Service layer communicates with the database and makes use of JPA to access data:*

Data storage and retrieval are managed by the persistence layer. It communicates with the database using tools like the Object-Relational Mapping (ORM) framework Hibernate or the Java Persistence API (JPA). The Persistence layer manages database transactions, carries out CRUD processes, and translates data between Java entities (objects) and database rows. By providing an abstraction over the underlying database system, it enables programmers to use entities and objects rather than simple SQL queries.

- *After the business processes are finished, the Controller sends the client a response, which might be in the form of a JSP page or another acceptable format:*

The Controller regains control once the Service layer has completed carrying out the required business operations. The Controller uses the request from the client and the application's settings to produce the response data, which can be in HTML, JSON, XML, or another format. For instance, if the client requested an HTML page, the Controller may deliver a JSP (JavaServer Pages) page or a Thymeleaf template created using data from the server. If the client requested JSON data, the Controller may serialize the response and deliver it in that format as the response body.

The thorough explanation of the flow of a typical Spring Boot application is now complete. With the Controller handling HTTP requests, the Service layer implementing the business logic, and the Persistence layer managing data storage and retrieval from the database, this approach enables the separation of responsibilities.

III. ADVANTAGES OF SPRING FRAMEWORK

Simplified Configuration: By offering default

configurations and auto-configuration based on classpath and convention, Spring Boot lowers boilerplate code and does away with the need for complex XML setups.

Rapid application development is made possible by Spring Boot's starting dependencies, which automatically establish common activities. This enables developers to quickly build up projects and concentrate on the logic of applications.

Embedded Server: Spring Boot comes with an embedded server (such as Tomcat or Jetty) that streamlines deployment by allowing standalone JAR files and straightforward distribution across various environments.

Opinionated Defaults: Spring Boot adheres to the principle of "convention over configuration," providing sane defaults and best practices to minimize the need for decision-making while still permitting modification.

smooth Integration: Spring Boot uses the extensive ecosystem of Spring libraries and extensions to develop scalable and reliable applications by integrating with other Spring projects in a smooth manner.

Support for testing: Spring Boot provides a thorough testing framework that makes it simple to write unit, integration, and end-to-end tests and includes tools for mocking and emulating dependencies.

Community and documentation: Spring Boot has a vibrant community with a wealth of tutorials, guides, and resources that offer a wealth of help, examples, and answers to common issues.

V. CONCLUSION

Java developers may construct business apps with the aid of the efficient Spring framework. With Spring Boot, it is easier for developers to get everything they need right away and concentrate just on the business logic. The embedded server and the absence of strict class connection make unit testing and module testing

easier. The Model View Controller and Dependency Injection components of a software simplify the developer's task.

VI. REFERENCES

[1] D. Lu, G. Mao, X. Wang and W. Tan, "A Research on Design of Campus Printing Service System," 2019 IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT), Harbin, China, 2019, pp. 183-186, doi: 10.1109/ICEICT.2019.8846321.

[2] Wali, Ravichandra & M, Prof. (2019). Rapid Web Application Development Using Spring Framework: A Case Study. International Journal of Innovative Research in Computer Science & Technology. 7.104-107.10.21276/ijrcst.2019.7.3.14.

[3] Singh, Dr. Tejinder. (2015). JAVA WEB DESIGN FRAMEWORKS: REVIEW OF JAVA FRAMEWORKS FOR WEB APPLICATIONS. IJARSE. 592-595.

[4] K. Guntupally, R. Devarakonda and K. Kehoe, "Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 5328-5329, doi: 10.1109/BigData.2018.8621924.

[5] Y. Gong, F. Gu, K. Chen and F. Wang, "The Architecture of Micro-services and the Separation of Front-end and Back-end Applied in a Campus Information System," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), Dalian, China, 2020, pp. 321-324, doi: 10.1109/AEECA49918.2020.9213662.

[6] Chavez, Alvaro & Wong, Lenis. (2022). EasyRest: Generador automático de API Rest basado en Spring Framework y motor de plantillas Beatl. Revista peruana de computación y sistemas. 4. 25-35. 10.15381/rpcs.v4i1.24125.

[7] Varanasi, Balaji & Bartkov, Maxim. (2022). Spring REST: Building Java Microservices and Cloud Applications. 10.1007/978-1-4842-7477-4.

[8] H. D and V. G. Charan, "e-seller Web Application for Vegetables using STS Framework," 2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICES), Chennai, India, 2022, pp. 1-6, doi: 10.1109/ICES55317.2022.9914380.

[9] Temirov, M.A. & Abykeev, K.Dj. (2022). THE ADVANTAGES OF DEVELOPING WEB APPLICATIONS USING THE SPRING MVC FRAMEWORK. The herald of KSUCTA n a N Isanov. 641-645. 10.35803/1694-5298.2022.2.641-645.

[10] Sun, Guiling & Wang, Yingjie & Li, Mengsha & Liu, Zhenjun. (2018). A Lightweight MVC Framework Based on Code Decoupling Principle. Journal of Computer and Communications. 06. 118-127. 10.4236/jcc.2018.63009.

[11] Liu, Na. (2023). Development of Industry-University-Research Institute Collaborative Innovation Information Platform Based on Spring MVC Framework. 10.1007/978-981-99-1157-8_12.

